

(12) UK Patent Application (19) GB (11) 2 323 000 (13) A

(43) Date of A Publication 09.09.1998

(21) Application No 9811995.1

(22) Date of Filing 15.04.1994

Date Lodged 04.06.1998

(62) Divided from Application No 9407526.4 under Section 15(4) of the Patents Act 1977

(71) Applicant(s)

VLSI Technology Inc
(Incorporated in USA - California)
1109 McKay Drive, M/S 45 San Jose,
California 95131, United States of America

Communicate Limited
(Incorporated in the United Kingdom)
The Technology Transfer Centre, Imperial College
of Science, Technology & Medicine, Silwood Park,
Buckhurst Road, ASCOT, Berkshire, SL5 7PW,
United Kingdom

(72) Inventor(s)

Ian Macaulay
Norman Hopkins

(51) INT CL⁶

H04L 29/06 , G06F 13/38

(52) UK CL (Edition P)

H4P PPEC

(56) Documents Cited

EP 0367284 A2

(58) Field of Search

UK CL (Edition P) H4P PPEC
INT CL⁶ G06F 13/38 13/42 , H04L 29/06
Online:- WPI

(72) cont

Brian C Daellenbach
Paul Denman
Jamie Osbourne

(74) Agent and/or Address for Service

Bowles Horton
Felden House, Dower Mews, High Street,
BERKHAMSTED, Herts, HP4 2BL, United Kingdom

(54) Abstract Title

Programmable serial interface

(57) A serial interface circuit 14 comprises a plurality of programmable channels (CHANNEL 0,1,2) which respond to a common bus 18. Each channel has a dedicated input/output (I/O) port 42 such that a multiplicity of ports are collectively capable of supporting a selected serial protocol (e.g. three-wire or two-wire) based on the programming of the channels. The programming takes the form of 8-bit data words provided by a microprocessor. The interface may be used as a bi-directional, parallel-to-serial and serial-to-parallel master circuit coupled between a microprocessor and a plurality of slave circuits (see fig. 1). The channels are programmed to operate as either a data channel or a clock channel.

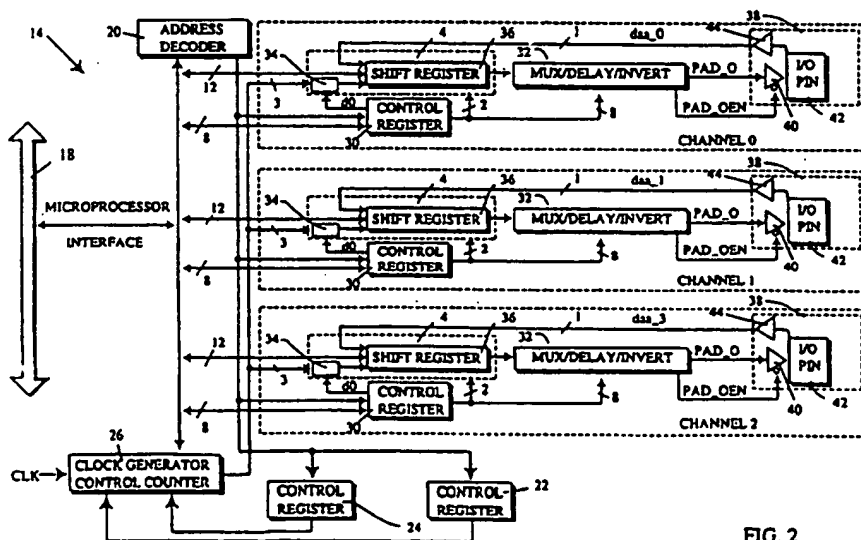


FIG. 2

BEST AVAILABLE COPY

GB 2 323 000 A

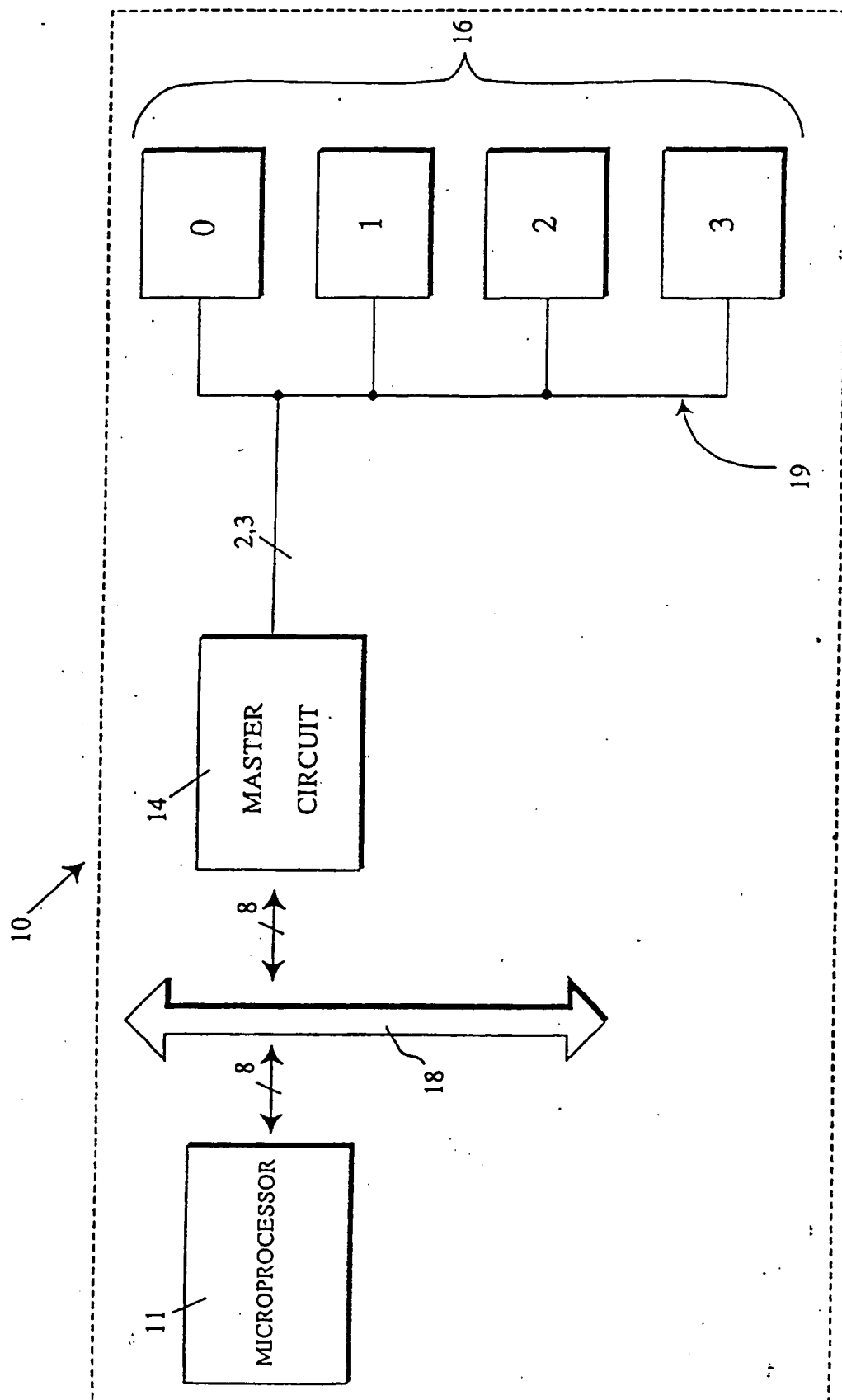


FIG. 1

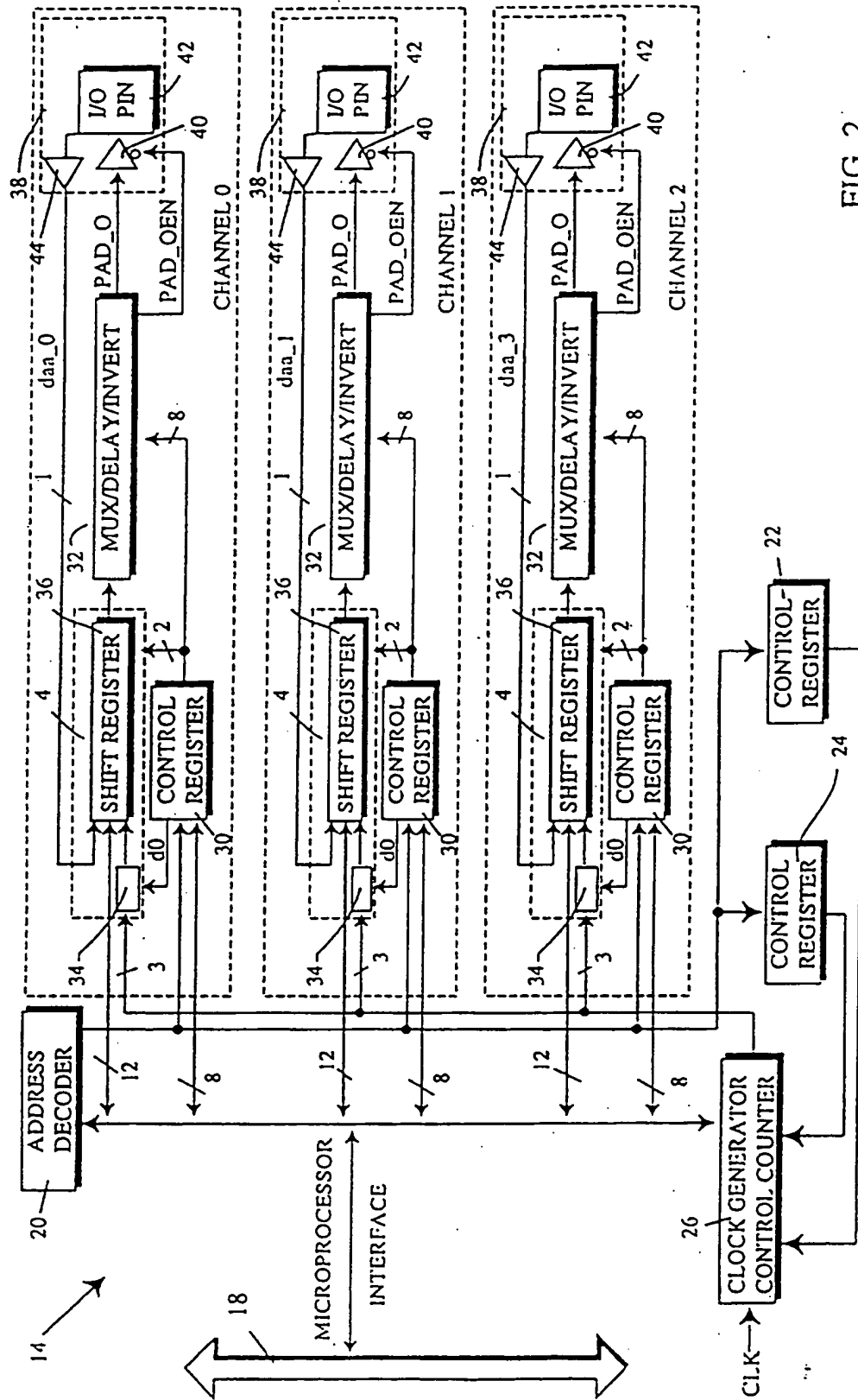


FIG. 2

30 →

d7	d6	d5	d4	d3	d2	d1	d0
TYPE-1	TYPE 0	INVERT OUTPUT	1/4 CYCLE	SR END	OPEN DRAIN	DIR	SR CLOCK

d7	d6	d5	d4	d3	d2	d1	d0
TYPE 1	TYPE 1	TYPE 0	INVERT OUTPUT	1/4 CYCLE	SHIFT REGISTER END	OPEN DRAIN	DIR
d6	TYPE 0	TYPE 1	TYPE 0	TYPE 1	TYPE 0	TYPE 1	TYPE 0
d5	INVERT OUTPUT	DATA OUTPUT TO PIN IS NOT INVERTED 0: DATA OUTPUT TO PIN IS NOT INVERTED 1: DATA OUTPUT TO PIN IS INVERTED	1/4 CYCLE	SHIFT REGISTER END	OPEN DRAIN	DIR	SR CLOCK
d4	1/4 CYCLE	NO DELAY ON DATA OUTPUT 0: NO DELAY ON DATA OUTPUT 1: A 1/4 CYCLE DELAY IS ADDED TO THE OUTPUT DATA	SHIFT REGISTER END	OPEN DRAIN	DIR	SR CLOCK	SR CLOCK
d3	SHIFT REGISTER END	SETS THE LOGIC VALUE WHICH WILL BE CLOCKED INTO THE SHIFT REGISTERS LSB DURING OPERATION.	OPEN DRAIN	DIR	SR CLOCK	SR CLOCK	SR CLOCK
d2	OPEN DRAIN	0: PUSH-PULL IF CHANNEL IS OUTPUT 1: OPEN-DRAIN IF CHANNEL IS OUTPUT	DIR	SR CLOCK	SR CLOCK	SR CLOCK	SR CLOCK
d1	DIR	0: CHANNEL IS INPUT 1: CHANNEL IS OUTPUT.	SR CLOCK	SR CLOCK	SR CLOCK	SR CLOCK	SR CLOCK
d0	SHIFT REGISTER CLOCK	0: RISING EDGE OF REFERENCE CLOCK USED TO CLOCK SHIFT REGISTER 1: FALLING EDGE OF REFERENCE CLOCK USED TO CLOCK SHIFT REGISTER	SR CLOCK	SR CLOCK	SR CLOCK	SR CLOCK	SR CLOCK

FIG. 3

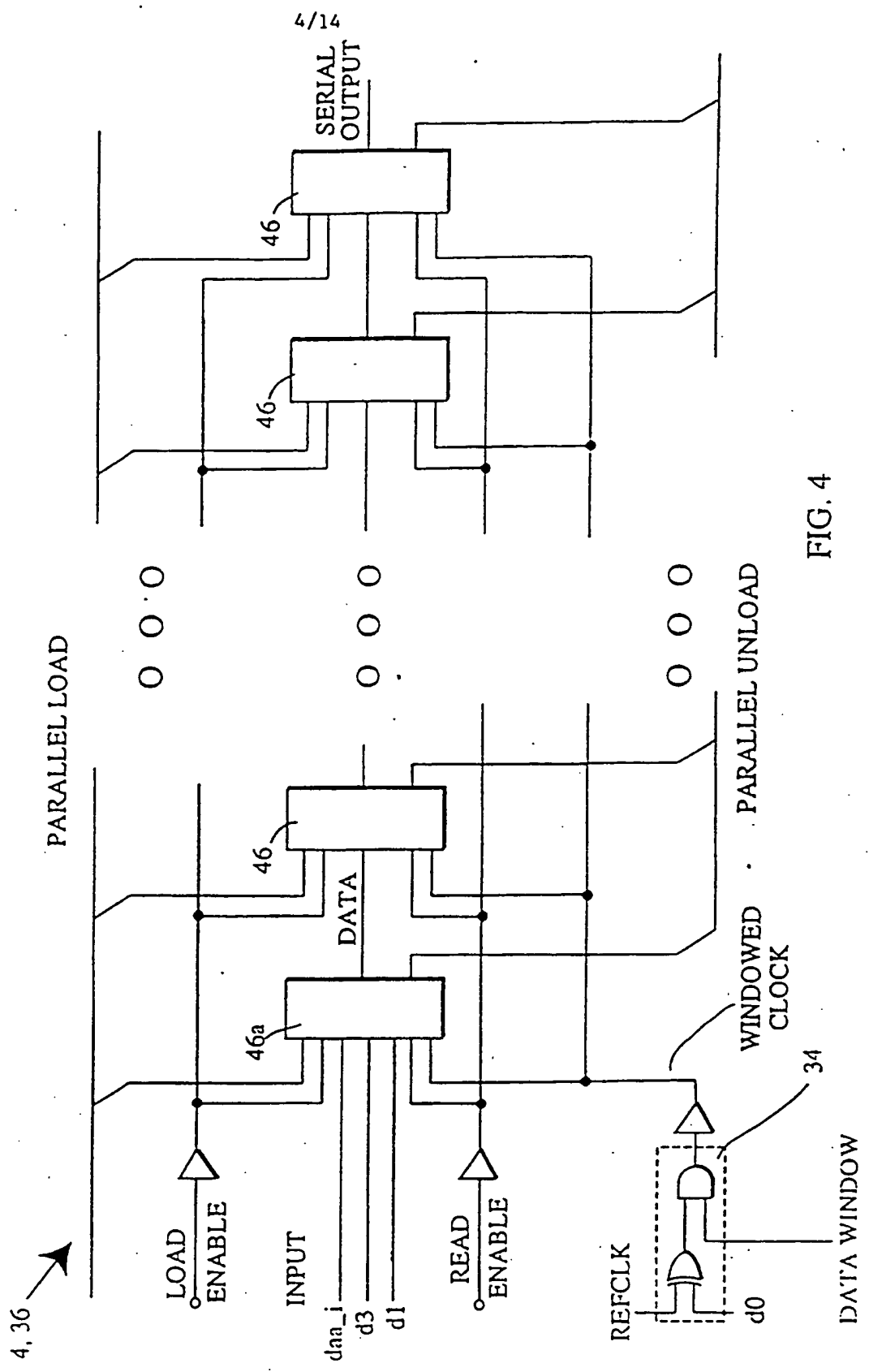


FIG. 4

FIG. 4a

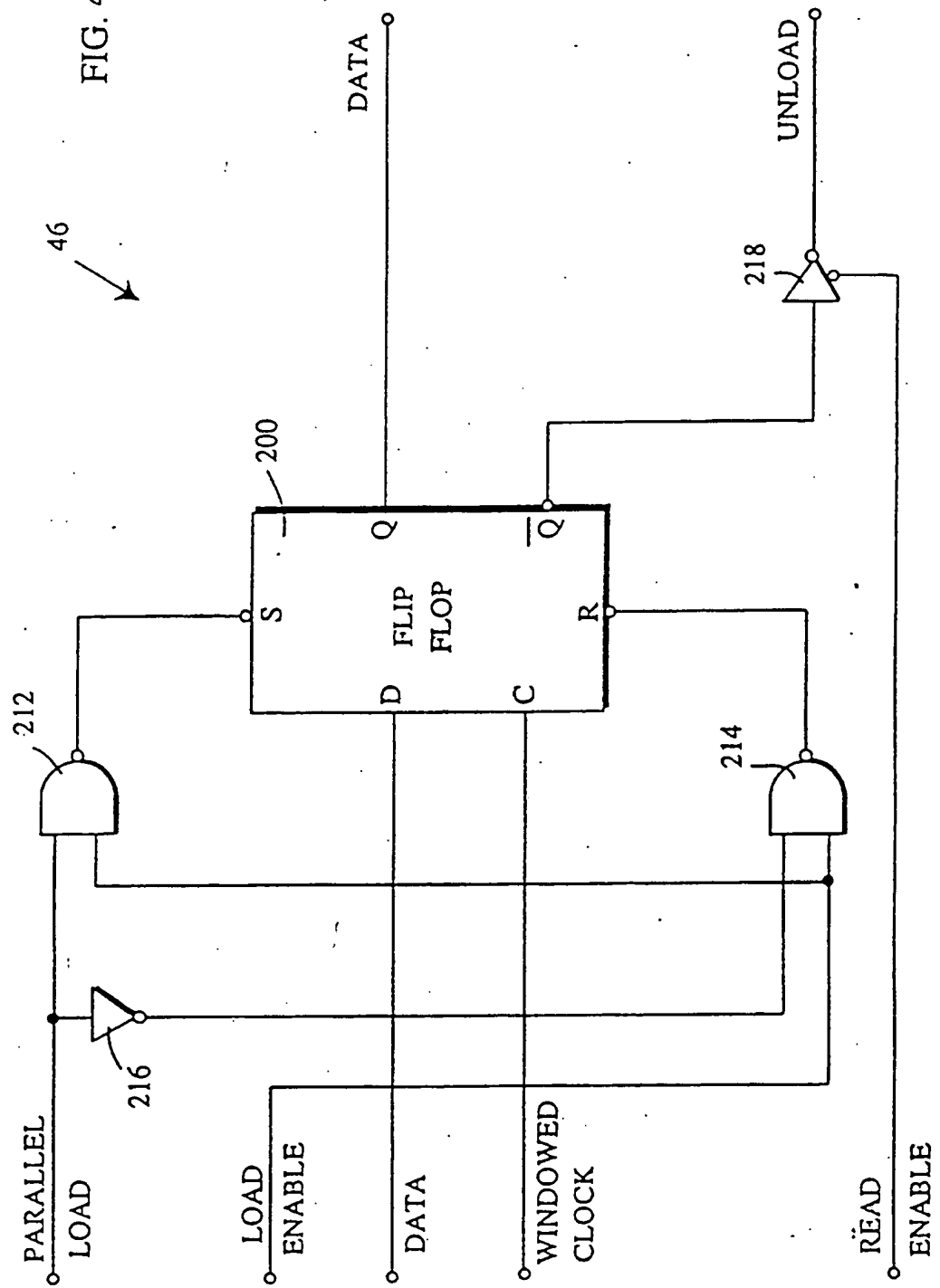
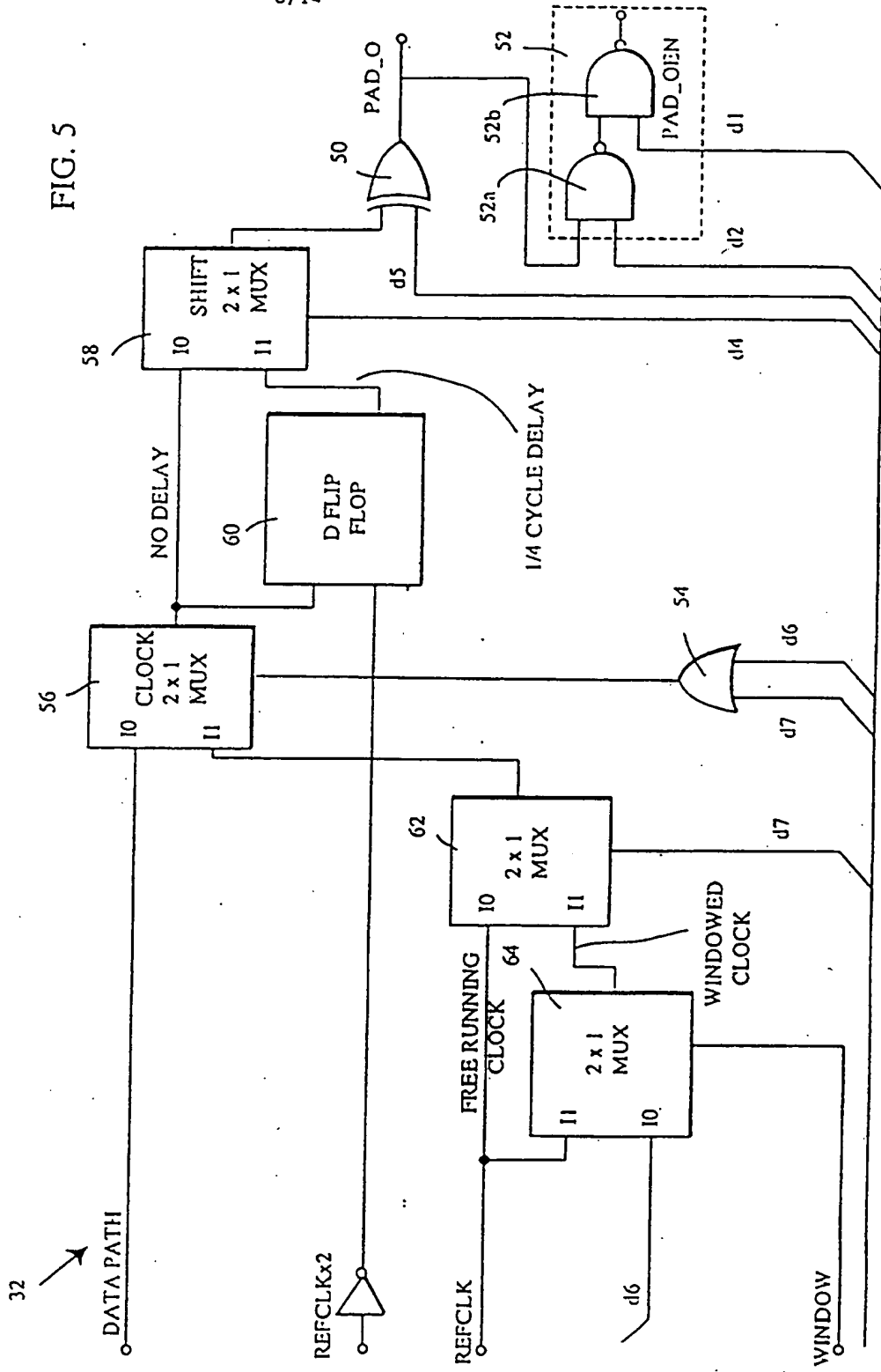


FIG. 5



d7	d6	d5	d4	d3	d2	d1	d0
NOT USED	NOT USED	SRC 1	SRC 0	B3	B2	B1	B0

SRC 1	TYPE 1	TYPE 0	REFERENCE CLOCK SOURCE
	0	0	INTERNAL RATE SELECTED ACCORDING TO B3, B2, B1, AND B0
SRC 0	0	1	daa_0 INPUT PIN
	1	0	daa_1 INPUT PIN
	1	1	daa_2 INPUT PIN
B3	THESE BITS SELECT A BIT RATE ACCORDING TO THE FREE RUNNING TIMER. THE ACTUAL RATE DEPENDS UPON THE CLOCK APPLIED TO THE SYSTEM.		
B2			
B1			
B0			

FIG. 6b

26 →

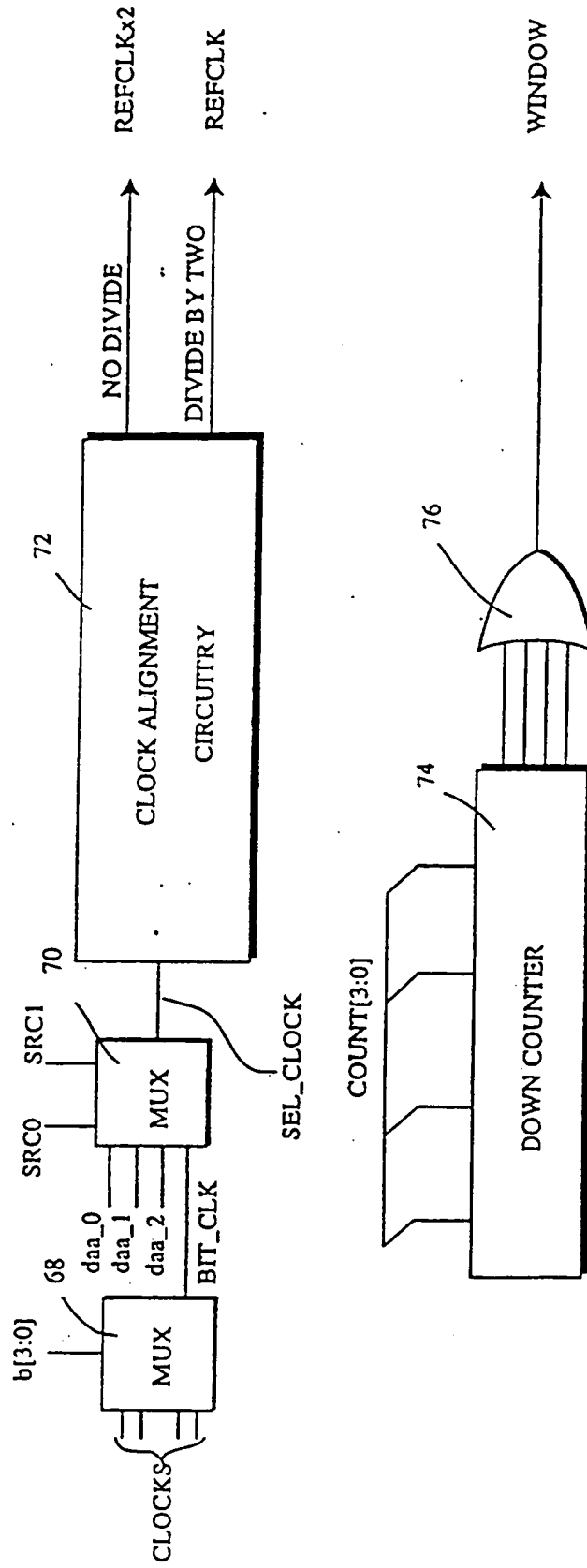


FIG. 7

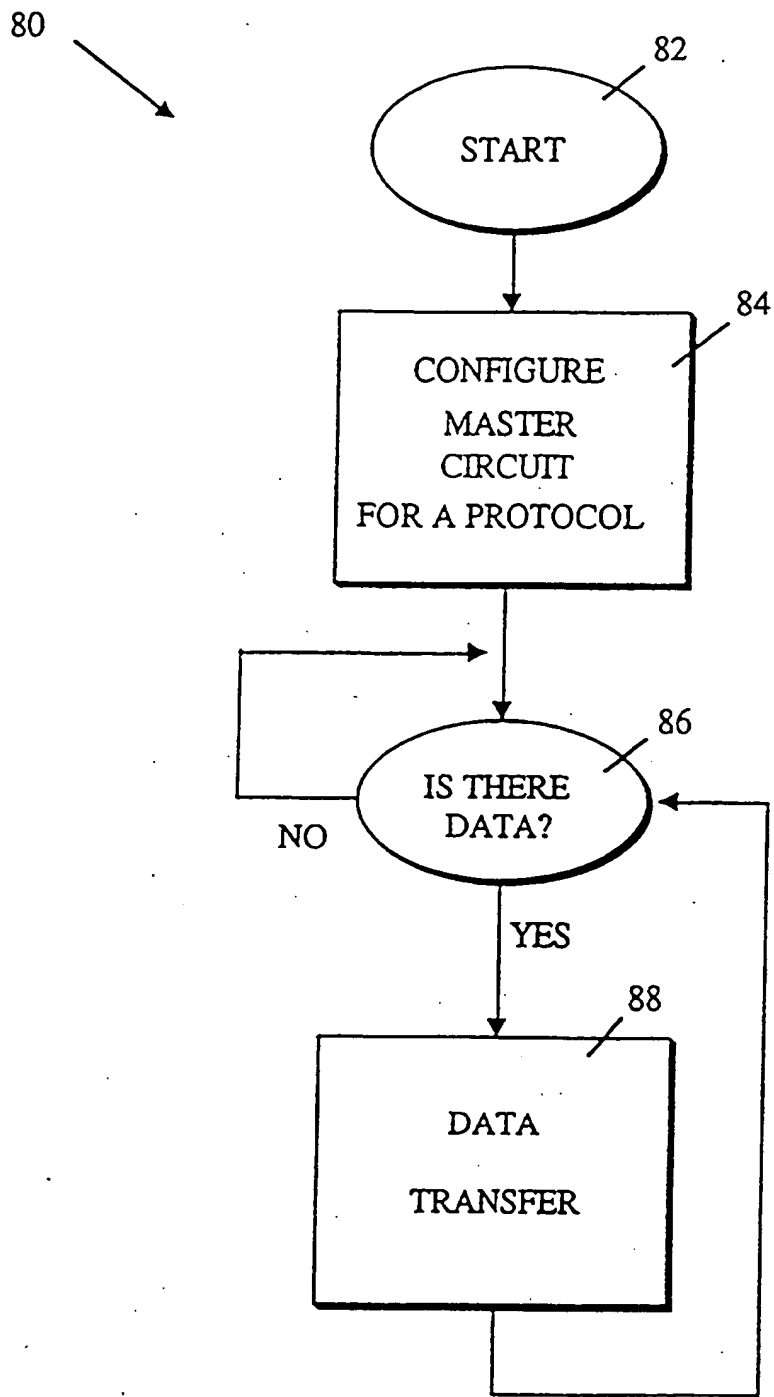


FIG. 8

88a

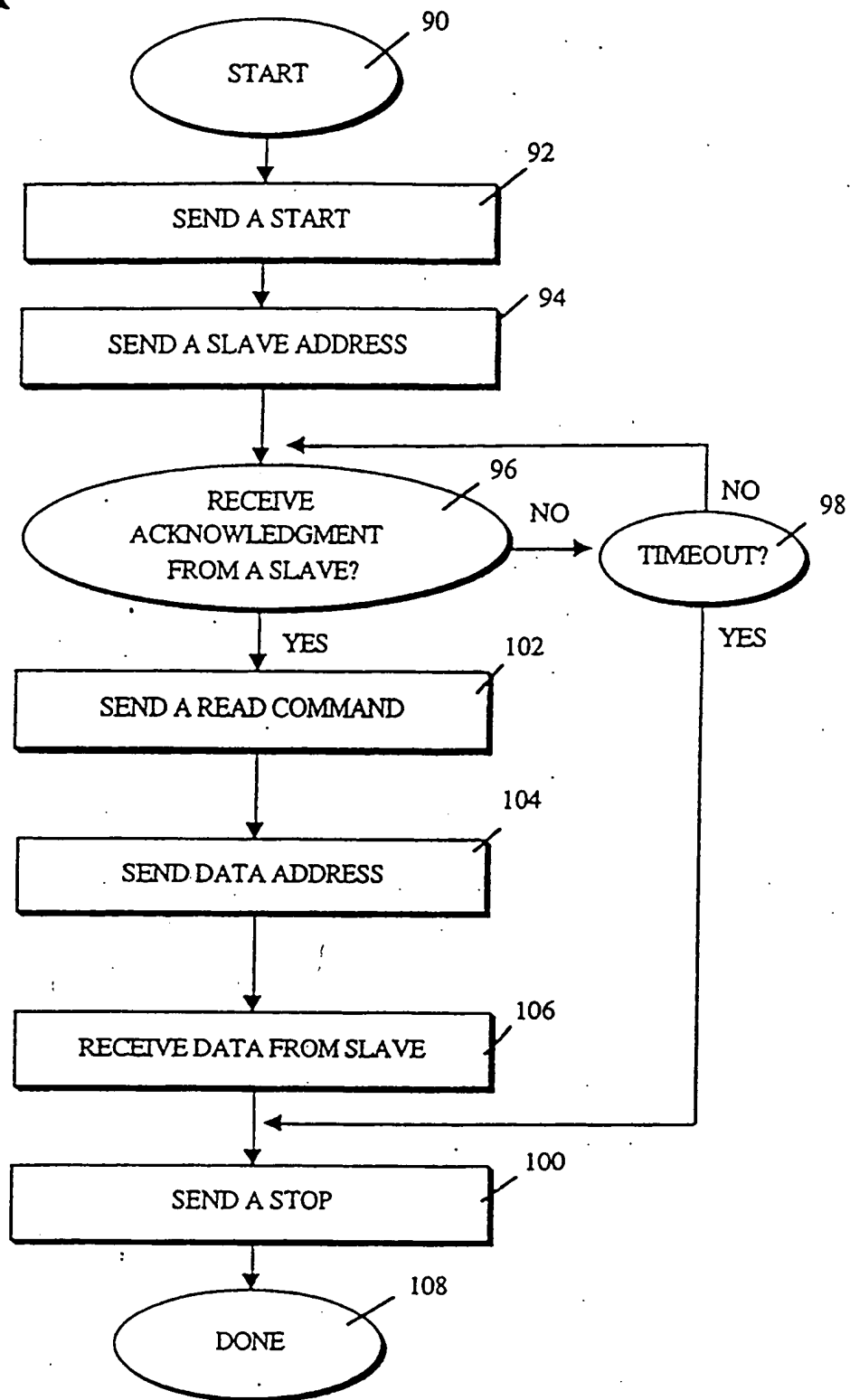


FIG. 9

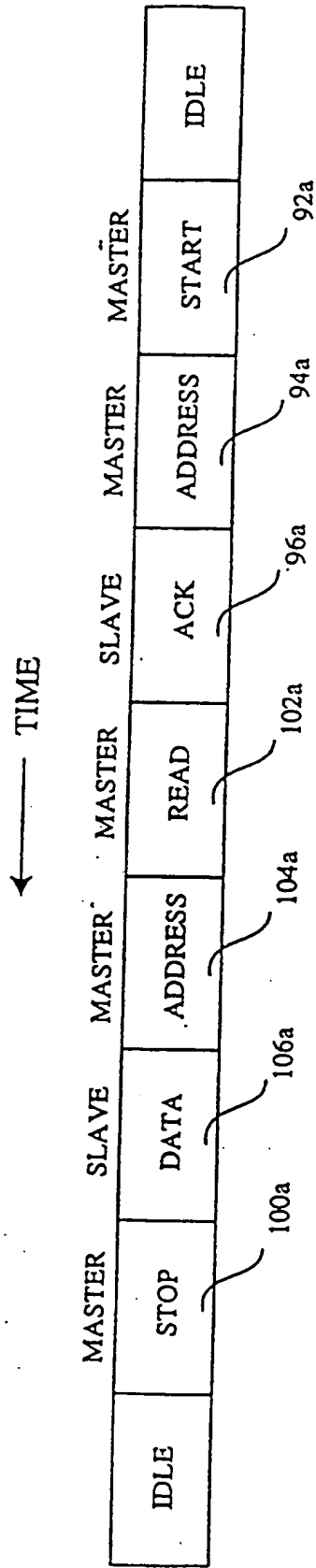


FIG. 9a

88b

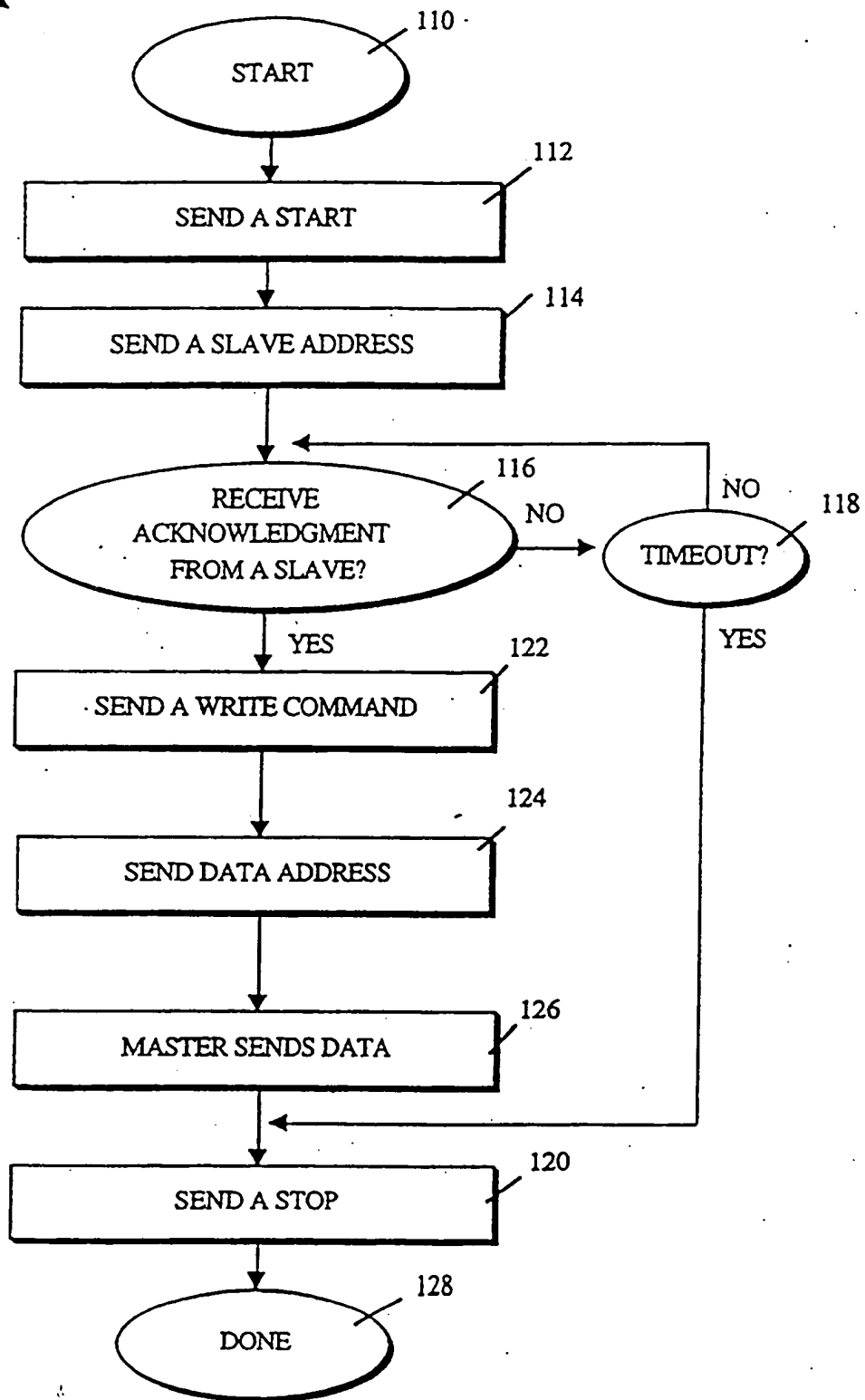


FIG. 10

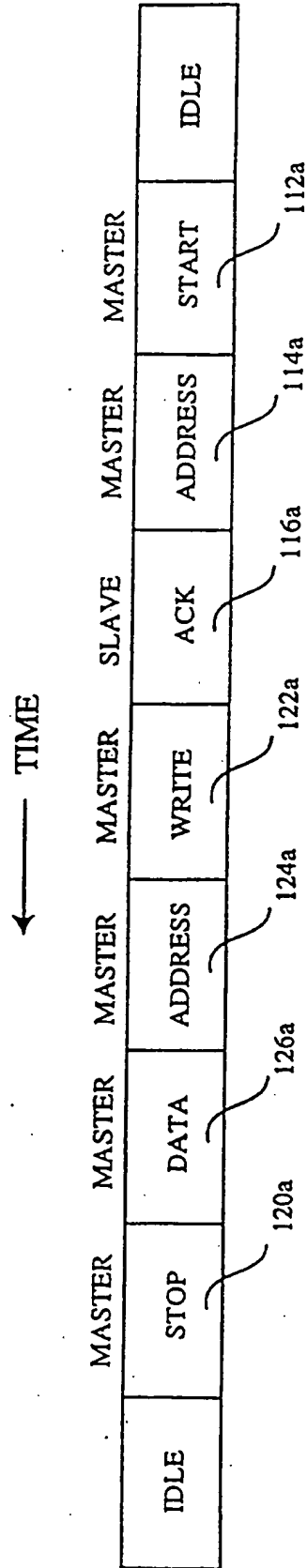


FIG. 10a.

SERIAL INTERFACE CIRCUIT

5

Description

Technical Field

This invention relates generally to electronic circuitry and more particularly to digital communication systems including data transfer protocols.

10 Background Art

Digital communication systems can be used to transfer data between digital devices. For a particular digital communication system, a "digital protocol" embodies the communications requirements and formats for data transfer. For example, a digital protocol can specify a method of "hand-shake" verification of a successful communications link between system elements. A data protocol can also detail the method for determining the direction of data flow during a data transfer and the types of data being transmitted. Circuits within a communications system must use the same digital protocol to prevent data loss or blockage.

Serial digital protocols are popular because they allow digital connections over a small number of wires. Several common serial digital protocols use either two or three wires. These protocols are therefore referred to as "two-wire protocols" and "three-wire protocols".

The current state of the art for transferring data in formatted character streams between a microprocessor and a "slave" circuit (i.e. a digital device controlled by the microprocessor) uses a hardwired serial interface circuit. Examples of communications systems using digital serial protocols are automotive braking systems which must communicate road information from each wheel to a master circuit and sensors at the joints of robot arms which must relay information to a processor to affect motion of the entire arm.

30 A widely used serial protocol is the National Semiconductor Microwire™ serial protocol. This three-wire protocol is typically built into hardware and cannot be changed or reconfigured after manufacture. Data is shifted in and out on a rising edge of a clock, and only master and slave circuits having digital controllers hard-wired with the same hand-shake protocol can be connected to each other.

35 Another widely-used serial protocol is the Phillips I²C™ serial protocol. This two-wire protocol is also typically implemented in hardware and thus cannot be

changed or reconfigured after manufacture. It uses a master clock produced by a master circuit to synchronize data transfers. These circuits are limited to communicating with other circuits utilizing the Phillips I²C™ serial protocol.

5 A problem with digital communications systems of the prior art is that they lack flexibility. In the past, a digital protocol was typically chosen by a system designer and was implemented in hardware. Therefore, for example, an end user would be unable to specify an I²C protocol if the system was hardwired with a Microwire protocol. Hardwired interfaces also prevent end users from creating their own, customized protocols.

10 It should be noted that a microprocessor coupled to, for example, a parallel port can be programmed to emulate in software a number of serial protocols. Such a solution would provide flexibility to the end-user, but is very wasteful of processing power. As a result, the use of a microprocessor to emulate in software a variety of serial protocols is not a practical solution to the aforementioned problem.

15 Disclosure of the Invention

The present invention solves these problems with a method and an apparatus which provides a digital communication system including a digital controller (typically a microprocessor), a master circuit capable of communicating in a variety of serial protocols, and at least one slave circuit. The master circuit is configured to a selected
20 serial protocol by the digital controller, and serves as a parallel-to-serial interface circuit between the digital controller and the slave circuit. The slave circuit is coupled to the master circuit and is responsive to communications in the master circuit's selected serial protocol.

In a preferred embodiment, the digital controller can configure various channels
25 of the master circuit to provide two-wire or three-wire serial protocols. The digital controller communicates with the master circuit with multi-bit words. The master circuit preferably includes a control register permitting the various channels of the master circuit to operate as either a data channel or a clock channel. If the channel is a clock channel, it can be either free-running or windowed. If the channel is a data
30 channel, the control register can select data inversion, delay, direction, and the triggering edge of an input clock.

The apparatus therefore preferably provides a serial interface circuit having several programmable channels responsive to a common address/data/control bus. Each channel has a dedicated input/output (I/O) port. Collectively, these I/O ports can
35 support a selected serial protocol based upon a programming of the channels. The

digital controller communicates with the master circuit with multi-bit words, thereby relieving the digital controller of direct bit-by-bit control of communications protocol. This greatly reduces the processing load on the digital controller.

5 A method of serial communication in accordance with the present invention is obtained by coupling a master circuit to a digital controller, coupling the master circuit to at least one slave circuit, and configuring the master circuit in a selected serial protocol. The master circuit communicates with the digital controller in multi-bit words and implements the selected serial protocol for the digital controller.

10 The present invention having a parallel-to-serial interface master circuit configurable in a wide variety of two-wire and three-wire protocols is a major advance over the previously described prior art. This flexibility frees circuit designers from the need to adhere to arbitrary proprietary standards and allows them to create communications systems particularly tailored to specific applications. Designers no longer need to be concerned with the compatibility of various components during
15 design because the configurable, flexible, interface automatically solves these problems. Indeed, the interface can communicate in one protocol to communicate with a select group of components and can then be reconfigured in another protocol to communicate with other components.

20 Previously, this flexibility could be implemented in software by the digital controller (i.e. the microprocessor) itself but only at a great processing burden to the digital controller. However, ability of the digital controller to communicate, in parallel, with the master circuit in full words rather than in a bit stream greatly reduces the communications burden on the digital controller.

25 Because of the extremely wide range of uses for computers, microprocessors, and computer networks, the flexible serial interface of the invention can be used in a multitude of devices. These and other advantages of the present invention will become apparent upon reading the following detailed descriptions and studying the various Figs. of the drawings.

Brief Description of the Drawings

30 Fig. 1 is a block diagram of a digital communication system in accordance with the present invention.

Fig. 2 is a schematic diagram of a master circuit of Fig. 1.

Fig. 3 is table showing the contents of a control register of Fig. 2.

Fig. 4 is a schematic diagram of a shift register circuitry of Fig. 2.

Fig. 4a is schematic of a flip flop used in the shift register of Fig. 4.

Fig. 5 is a schematic diagram of a multiplexer/delay/inverter circuit of Fig. 2.

Fig. 6a illustrates the contents of a counter control register.

5 Fig. 6b illustrates the contents of a clock control register.

Fig. 7 is a diagram of a clock generator/control counter circuit.

Fig. 8 is a flowchart showing a basic sequence of steps forming a method of serial communication in accordance with the present invention.

10 Fig. 9 is a flowchart showing a sequence in a step of data transfer in Fig. 8 to read from a slave circuit.

Fig. 9a illustrates a sequence of commands exchanged between a master circuit and a slave circuit to read data from a slave circuit during the data transfer of Fig. 9.

Fig. 10 is a flowchart showing a sequence in a step of data transfer in Fig. 8 to write data to a slave circuit.

15 Fig. 10a illustrates a sequence of commands exchanged between a master circuit and a slave circuit to write data to a slave circuit during the data transfer of Fig. 10.

Best Modes for Carrying out the Invention

20 In Fig. 1, a preferred embodiment of a digital communication system 10 in accordance with the present invention includes a digital controller 11, master circuit 14, and one or more slave circuits 16. The digital controller 11, the master circuit 14, and the slave circuits 16 are all digital devices operating with digital logic states. These logic states will be referred to herein as variously 0, low, and LO for the logic state "0", and 1, high, and HI for the logic state "1." By convention, a LO logic state is often at about 0 volts d.c., and a HI logic state is often at about 5 volts
25 d.c. Another convention uses 0 volts d.c. and 3 volts d.c. as representing LO and HI logic states. Still other conventions can be chosen.

30 The digital controller 11 is typically a microprocessor, such as an ARM microprocessor made by VLSI Technology, Inc. and Sharp, Inc., and others. Alternatively, the microprocessor could be an Intel x86 compatible microprocessor or a Motorola 680X0 compatible microprocessor, for example. The digital

controller 11 can also be implemented as microcontroller or a state machine, for example. Preferably, however, the digital controller 11 is a microprocessor that is programmable to run a control process which configures master circuit 14 and causes data transfer between the digital controller 11 and the several slave circuits

5 16.

Slave circuits 16 are typically provided with unique addresses. In this example the addresses of the four slave circuits are 0, 1, 2, and 3, respectively. The master circuit 14 is coupled to the slave circuits 16 by a slave bus 19 which, in this preferred embodiment, can be either 2 or 3 bits wide. The digital controller 11 is coupled to the master circuit by a standard bi-directional microprocessor bus 18. Bus 18 generally includes address, data, chip select (CS), and read/write (R/W) lines.

By sending words to the master circuit 14, the digital controller can configure the master circuit 16 to have either a two-wire or three-wire serial protocol for sending a bit stream to the slave circuits 16. The size (i.e. number of bits) of the data words can be selected for a particular communications system. The preferred embodiment discussed here will refer to 8-bit data words. As used herein, a "word" is a multi-bit unit of data, provided simultaneously and in parallel. Sometimes, those skilled in the art will refer to a word, such as the 8-bit word described herein, as a "character", since alphanumeric characters can be encoded in 8-bit words.

Once the master circuit 14 has been configured by the process implemented by the controller 11, the master circuit can act as a serial communications device to communicate in a selected serial protocol. As mentioned previously, the bit stream between the master circuit and the slave circuits is then either in a two-wire or a three-wire protocol in this preferred embodiment. However, other protocols are also possible.

A block schematic diagram of a representative master circuit 14 appears in Fig. 2. As previously stated, the interface between the master circuit 14 and the digital controller 11 is preferably through a bi-directional microprocessor bus 18. Data words sent to the master circuit 14 from the digital controller 11 are applied to the inputs of an address decoder 20 which directs the data word to one of five control registers, such as the three control registers 30, or the control registers 22 or 24.

The two-wire or three-wire bit streams are transferred between the master circuit 14 and the slave circuits 16 correspond to two channels or three channels of the master circuit 14 shown in Fig. 2. The three channels are labeled "CHANNEL 0", "CHANNEL 1", AND "CHANNEL 2."

5 Before data can be transferred between the master circuit 14 and a slave circuit 16, the master circuit 14 is preferably configured by programming the five control registers. A separate data word is used for each control register. Therefore, five data words are used to configure the master circuit 14. However, no particular order of programming these registers is required.

1 0 A combined clock generator/control counter 26 processes an incoming clock signal to develop a reference clock REFCLK, a double frequency clock REFCLKx2 clocking at twice the frequency of REFCLK, and a data window. To configure the control counter 26, the digital controller 11 sends a data word to the master circuit 14. The address decoder 20 decodes the address of a counter control register 22 and enables the counter control register 22 to accept the data word. This register will be described in the discussion of Fig. 6a. To configure the clock generator 26, the digital controller 11 sends another data word to the master circuit 14. The address decoder 20 decodes the address of a clock control register 24 (which is further described with reference to Fig. 6b) and enables the clock control register 24 to accept the data word. The counter control register 22 and the clock control register 24 together configure the combined clock generator/control counter 26.

2 5 The step of configuring a channel of the master circuit 14 will only be described in detail for a single representative channel in which like reference numerals refer to like components. This step is done once per channel for a total of three times. The address decoder 20 enables a control register 30 for a channel selected by the digital controller 11. The control register determines the mode of operation of the channel by configuring the shift circuitry 4 and the MUX/delay/inverter circuitry 32 of the selected channel.

3 0 The shift circuitry 4 contains a clock circuit 34 and a shift register 36. In an "output" mode corresponding to the sending of data from master circuit 14 to a slave circuit 16, the shift register 36 acts as a parallel-to-serial interface; parallel data is input to the shift register 36 from the digital controller 11 and the data is serially shifted out through the MUX/delay/inverter 32 to an output buffer 40. Output buffer 40 is enabled by a PAD_OEN signal to couple the signal PAD_O developed

by the MUX/delay/inverter 32 to an I/O pin 42. A more detailed schematic of the MUX/delay/inverter 32 is shown in Fig. 5 and will be described below.

In an "input" mode corresponding to the master circuit 14 receiving data from a slave circuit 16, data is received by an input buffer 44. This buffer applies a signal daa_0, daa_1, or daa_2 (for channels 0, 1, and 2, respectively) to the corresponding shift register 36 where data is then subsequently transferred to the digital controller 11. In this mode of operation, the shift register 36 functions as a serial-to-parallel interface by serially shifting data in from the slave circuits and providing a parallel output word for the digital controller 11.

Fig. 3 is a table illustrating the contents and programming features of the control registers 30. As stated above, a preferred embodiment is to have eight bit data words for programming the control register. The bits are denoted by d0 through d7. The d0 bit of the control register 30 configures the clock circuit 34 to determine whether the rising or falling edge of the reference clock is used to clock the shift register 36. The d1 through d7 bits configure the shift register 36 and the MUX/delay/inverter 32 and further specify the function of the input/out (I/O) pad 38.

The truth table shown for bits d6 and d7 selects different clocks for the shift register 36. If a window is asserted, the data in the shift register is clocked. If the window is not asserted, the shift register is not clocked. As used herein, "high" shall mean bit d7 = 1 and "low" shall mean d7 = 0. "Low" shall mean d6 = 0 and "high" shall mean d6 = 1. For d7 high, a windowed clock is provided instead of a free-running reference clock. The window signal is produced by the clock generator circuit 26. Bit d6 provides additional clock selection.

Bits d1 and d3 are the bits of the control register 30 which program the shift register. Bit d3 determines the idle value being clocked into the shift registers 36. The d1 or DIR bit configures the direction of data. As in the discussion of Fig. 2, output corresponds to data being sent to the slave circuit 16 from the digital controller 11 in a selected serial protocol. Data input corresponds to receiving data into the master circuit 14 from a slave circuit 16 in a serial fashion and forwarding the data in a parallel fashion to the digital controller 11. Bits d4 and d5 set output delay and output inversion.

The d2 bit specifies whether the channel output is in an open drain or push-pull signal. The open drain arrangement allows the particular channel to share an external line with another signal source. With an open drain, the output buffer

drives when the output is low, but not when it is high. In the push-pull arrangement, no external line is shared and buffer 40 is always driven either high or low. The logic circuit of the channel simulating this arrangement with external circuitry is described in Fig. 5.

5 A schematic diagram of the shift circuitry 4, including the shift register 36 and the clock circuit 34, is shown in Fig. 4. The shift register 36 has a flip-flop 46 for each bit of the parallel load. In the preferred embodiment under discussion, there are twelve such flip-flops. The three inputs daa_i, d3, and d1 have already been described where daa_i where {i: 0, 1, 2} and thus is shorthand for the three
10 slave inputs daa_0, daa_1, and daa_2, respectively. The input flip flop 46a possesses additional logic to determine its input source. In output mode, the shift register 36 sequentially shifts the parallel load to a serial output. In input mode, data daa_i in channel i {i: 0, 1, 2} from a slave circuit 16 is sequentially loaded into the shift register 36 where it is output in parallel. The REFCLK and data window
15 inputs to the shift clock register 34 are from the clock generator/control counter 26. The windowed clock output of the shift clock register 34 is input to the control pins of the flip-flops 46. Effectively, the reference clock is gated by the window to clock the flip-flops.

A typical schematic of the flip-flop 46 is shown in Fig. 4a. A standard
20 edge-triggered flip-flop 200 is loaded with parallel input via a set NAND gate 212 and a reset NAND gate 214. This controlled by the load enable signal. The windowed clock from the clock circuit 34 is connected to the clock input C of the flip-flop. Data can be sequentially serially shifted from flip-flop to adjacent flip-flop, i.e. from the Q output of a first flip-flop to the D input of a second flip-flop,
25 etc.

A parallel unload can occur when an output buffer 218 is read enabled. This occurs when the shift register 36 acts as a serial-to-parallel interface to unload the stored values of the shift register flip-flops 200 after a serial input from a slave circuit 16 has been sequentially clocked into the shift register 36. Other
30 configurations of conventional flip-flops and logic gates can produce the flip flop 46 performing the same functions in the shift register 36.

The programming functions of the bits of control register 30 within the MUX/delay/inverter 32 are shown in Fig. 5. Bit d1 of control register 30 configures the direction of data flow by enabling the pad to be an output or an
35 input. XOR (exclusive OR) gate 50 causes the output to be inverted or not inverted, depending upon the state of d5. The NAND gates 52a and 52b allow the

circuit to form open drain logic 52 as controlled by d2. The open drain logic forms a full open drain circuit when combined with output buffer 40 (see Fig. 2). In open drain mode, the circuit enables the output buffer 40 when the output is LO, and disables it when the output is HI. In push-pull mode, the output buffer 40 is
 5 always enabled.

The OR gate 54 uses the d6 and d7 bits to determine whether the channel is a clock channel or a data channel. The 2-to-1 clock multiplexer 56 selects the data signal from the upper path when the output of the OR gate 54 is LO and selects the clock signal from the lower path when the output of the OR gate 54 is HI. The 2-
 10 to-1 shift multiplexer 58 chooses output directly from the clock multiplexer 56 when d4 is LO or output from the D flip-flop 60 when d5 is HI, which provides a quarter cycle data delay.

The clocks applied to the clock multiplexer 56 are controlled by the register bits d6 and d7 and the signal WINDOW. When d7 = 0, the output of the 2x1
 15 MUX 62 is the free running clock REFCLK. When d7=1, the output of the 2x1 MUX 62 is a windowed clock produced by the 2x1 MUX 64. When the signal WINDOW is HI, the windowed clock is derived from REFCLK, and when the signal WINDOW is LO, the windowed clock is derived from the bit d6.

Fig. 6a shows the contents of a counter control register 22. Only the first
 20 four bits are used by the window counter in Fig. 7. These bits are input to the window counter portion of circuit 26, which comprises a down counter. This register 22 represents the starting value of the counter.

As seen in Fig. 6b, only the first six bits of the data word are used in configuring the clock generator register in this preferred embodiment. They
 25 determine the bit rate by selecting a reference clock.

Fig. 7 illustrates the clock generator/control counter 26. The clock generator part of Fig. 7 selects an input clock from any of a number of system sources using a clock multiplexer 68 switched by bits b0 - b3. The resulting BIT_CLK signal is applied to a clock select multiplexer 70 where the SRC0 and
 30 SRC1 data word bits determine the clock used based on the truth table in Fig. 6b. The resulting SEL_CLOCK is input to the alignment circuitry 72 to develop the reference clocks REFCLK and REFCLKx2 where REFCLKx2 has twice the frequency of REFCLK.

The bottom four bits (i.e. COUNT 0 - COUNT 3) of the counter control
 35 register 24 shown in Fig. 6a determine the WINDOW signal. WINDOW is

asserted as long as the count is non-zero and WINDOW is de-asserted when the count reaches zero. This is accomplished by coupling all of the bits of the down counter 74 to the inputs of an OR gate 76 and thus asserting the window when any of bit is HI.

5 A method 80 for serial communication is illustrated in Fig. 8. The first step 84 configures the master circuit 14 for a selected serial protocol. If there is no data to transfer, step 86 enters an idle state awaiting data. Otherwise, data transfer occurs in step 88, after which process control returns to step 86. The steps associated with data transfer are illustrated more explicitly in Figs. 9 and 10.

10 The data transfer method 88a for reading data from a slave 16 is illustrated in Fig. 9. The master circuit 14 is operating under the direction of a process running on the digital controller 11. The process begins at 90 and, in a step 92, the digital controller 11 (typically a microprocessor) sends a start command to the master circuit 14 followed by a slave address (such as 0-3) in a step 94. The addressed slave circuit 15 16 will then send an acknowledgment or confirmation back in a step 96 to the master circuit 14 and controller 11 if it is ready to transfer data. If the acknowledgment is not sent within a specified time, a "time-out" decision is made in a step 98 whether the master circuit 14 and digital controller 11 should continue to wait for a response or whether the digital controller 11 should send a stop command in a step 100. Assuming 20 an acknowledgment is sent by the slave 16 and received by the master 14 and controller 11 within the time-out period, a read command is then sent to the slave in a step 102 followed by a data address in a step 104. The slave circuit responds in a step 106 by sending data to the address sent in step 94. The data transfer ends in a step 108 when the master circuit 14 sends a stop command in a step 100 to the slave circuit 16.

25 Fig. 9a shows a sequence of data transferred between the master circuit 14 and a selected slave circuit 16 during a successful data transfer process 88a for reading data. The sequence begins at the right of the diagram and the time sequence is right-to-left. The data corresponding to the steps in process 88a in Fig. 9 are labeled with the same number with the addition of a suffix "a." Initially, the system is in a idle state until data 30 transfer step 88a begins. The master circuit 14 first transmits a start command to a selected slave circuit 16 followed by slave circuit address. The selected slave circuit 16 responds with a primary acknowledgment. This initial acknowledgment is required for the commencement of data transfer. Subsequent acknowledgments (not shown) from the selected slave circuit 16 to the commands from the master circuit 14 occur after each 35 transfer (as will be appreciated by those skilled in the art), but are not included here for simplicity. The master circuit then sends a read command to the selected slave circuit followed by the data address. The slave then sends the data. Finally, the master circuit

16 sends a stop command to the selected slave circuit, and the system returns to its idle state.

5 The related data transfer process 88b for writing data to a slave circuit 16 is illustrated in Figs. 10 and 10a. Again, the master circuit 14 is operating controlled by a process running on the digital controller 11. The process begins at 110 and, in a step 112, the digital controller 11 sends a start command to the master circuit 14 followed by a slave address in a step 114 (such as 0-3). The master circuit 14 and digital controller 11 should then receive an acknowledgment in a step 116 from the slave circuit 16. If the acknowledgment is not sent within a specified time, a decision is made in a step 118
10 whether the master circuit 14 and digital controller 11 should continue to wait for a response or end a stop in a step 120 (i.e. "time-out"). Assuming an acknowledgment is sent by the slave 16 and received by the master 14 and controller 11, a write command is sent in a step 122 to the addressed slave circuit 16 followed by a data address 124. The master circuit 14, in a step 126, sends data to the data address specified in step
15 124. The data transfer ends with a step 128 when the master circuit 14 sends a stop command in a step 120 to the slave circuit 16.

Fig. 10a shows a sequence of commands transferred between the master circuit 14 and a selected slave circuit 16 during a successful data transfer 88b for the process of writing data 110. The data corresponding to steps in process 88b are labeled with
20 the same numbers with an "a" suffix. The process begins at the right of the diagram, and the time sequence is right-to-left. Initially, the system is in an idle state until data transfer 88 begins. The master circuit 14 first transmits a start command to a selected slave circuit 16 followed by slave circuit address. The selected slave circuit 16 sends an acknowledgment. As in the process of reading from a slave, this initial
25 acknowledgment is required to initiate the transfer. Subsequent acknowledgments (not shown) from the selected slave circuit 16 to the commands from the master circuit 14 are not shown, for simplicity. The master circuit then sends a write command to the selected slave circuit followed by the data address. Finally, the master circuit 16 sends a stop command to the selected slave circuit, and the digital communications system
30 returns to its idle state.

The master circuit 14 is configurable in wide variety of serial interface protocols including, but by no means limited to, the Phillips I²C and Microwire. Therefore, the method of serial communications 80 can be defined by I²C, Microwire, other standardized protocols, and custom designed protocols.

CLAIMS

1. A serial interface circuit comprising:

a plurality of programmable channels responsive to a common address/data/control bus, where each channel has a dedicated I/O port such that a multiplicity of said dedicated I/O ports are collectively capable of supporting a selected serial protocol based upon a programming of said plurality of programmable channels.
2. A serial interface circuit as recited in claim 1 wherein each of said plurality of programmable channels includes a control register which can be programmed over said address/data/control bus.
3. A serial interface circuit as recited in claim 2 wherein said multiplicity of dedicated I/O ports can support a three-wire serial protocol.
4. A serial interface circuit as recited in claim 2 wherein said multiplicity of dedicated I/O ports can support a two-wire serial protocol.
5. A serial interface as recited in claim 4 wherein said multiplicity of dedicated I/O ports can support a three-wire protocol.
6. A serial interface circuit as recited in claim 2 wherein each of said programmable channels includes: a control register coupled to said address/data/control bus; a shift register coupled to said address/data/control bus and to said control register; and protocol circuitry coupled to said control register and between said shift register and said I/O port.
7. A serial interface circuit as recited in claim 6 further comprising an address decoder coupled to said address/data/control bus and operative to selectively enable a shift register and a control register of a selected programmable channel.

8. A serial interface circuit as recited in claim 6 wherein said control register includes a plurality of control bits, wherein at least one of said control bits causes said programmable channel to operate as a data channel or a clock channel.
9. A serial interface as recited in claim 8 wherein at least one control bit causes said programmable channel to operate as either a free-running clock or a windowed clock when operating as a clock channel.
10. A serial interface circuit as recited in claim 8 wherein, when said programmable channel is operating as a data channel, bits of said control register correspond to: (a) data inversion; (b) data delay; (c) data direction; and (d) the triggering edge of an input clock which clocks said data channel.